MMoE

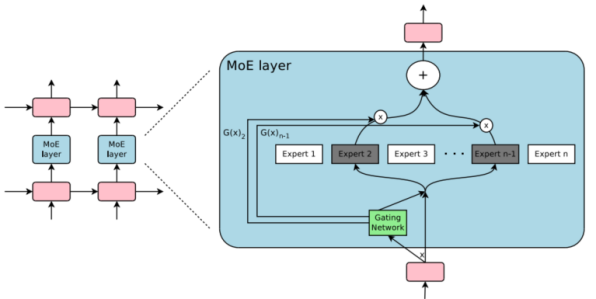
2018年谷歌团队发表了论文《Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts》，提出了一种新的多任务学习方法，将这种结构命名为多门专家混合网络Multi-gate Mixture-of-Experts，该结构使用混合专家结构，通过在所有任务中共享专家子模型，同时还拥有一个经过训练的门控网络来优化每个任务。

混合专家网络即MoE模型，是17年谷歌团队在论文《[Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer](https://lumingdong.cn/go/tyrcm0" \t "https://lumingdong.cn/_blank)》中提出的结构，是一种特殊的神经网络结构层，结合了专家系统和集成思想。虽然MoE最初是作为多个独立模型的集成方法开发的，但有部分研究员将其转化为基本的构建块MoE层，并将其堆叠在DNN中，得到了广泛使用。MoE层具有与MoE模型相同的结构，接受前一层的输出作为输入，并将其输出到后续层。然后以端到端方式训练整个模型。刚开始MoE层结构的主要目标是实现条件计算。

大模型一般能都取得很好的效果，一般来说，模型越大，效果越好，但是会面临很多的问题，比如模型训练代价高，效率低，存储空间大，通信代价高等，MoE是一种解决方法，MoE可以动态的激活神经网络，从而实现不增加计算量的前提下，大幅度的增加模型的参数量，MoE是现在训练万亿级别参数量的关键技术。本质上属于高效的路由方法，主要由专家划分和路由选择两部分构成，专家可以看成是具有部分参数的前馈神经网络，该门控网络选择专家的一个稀疏组合来处理每个输入，它可以实现自动分配参数以捕获多个任务可共享的信息或是特定于某个任务的信息，而无需为每个任务添加很多新参数，而且网络的所有部分都可以通过反向传播一起训练。当门控网络输出的值为非零即一时，鼓励专家之间处于竞争关系，而当输出值为小于一的权重向量时，鼓励专家之间处于合作关系。简单来说，MoE的思想是训练多个神经网络，也就是多个专家，每个专家通过门控网络被指定应用于数据集的不同部分，最后再通过门控网络将多个专家的结果进行组合。单个模型往往善于处理一部分数据，不擅长处理另外一部分数据，而多专家系统则很好的解决了这个问题：系统中的每一个神经网络，也就是每一个专家都会有一个擅长的数据区域，在这组区域上该专家就是权威，要比其他专家表现得好。因此多专家系统是单一全局模型或者多个局部模型的一个很好的折中，这样的网络结构能够处理更加复杂的数据分布，在相应的任务中，性能也会有很大的提升。

MoE采用分治法思想，将大的问题拆分成各个小问题，训练不同的专家来针对性地解决这些小问题，最后再将专家们的输出结合起来。MoE可以作为子层，嵌入到网络模型中，常见的为嵌入到LSTM、CNN网络中。对于MoE的研究点，一般也集中于专家划分和路由选择这两个部分，在专家划分方面，研究点为如何划分专家，划分专家的数量，专家的维度，如何初始化等方面，专家的数量可以为子任务的个数，初始化时可以选择随机初始化，或者依经验初始化等。在路由选择方面，研究点为输出的值为哪种类型，每个专家的权重为多少等内容。MoE的应用有视觉领域提出的V-MoE，多领域提出的LI-MoE等。

MoE模型虽然能够简化计算，增大参数量，但是会带来其他问题，负载不平衡，MoE层的稀疏程度， MoE层的建模与优化，MoE层的并行计算效率等问题。其中，负载不平衡问题被大家广泛关注，训练MoE模型的过程中，在最初的几个样本上表现较好的专家会被分配一个更大门函数输出，进而较大的门函数输出使得其对应的专家被更频繁地选择，并得到更充分的优化，这使得MoE模型中只有部分专家得到充分的优化，而其余专家无法被充分训练，这就是负载不平衡问题。在深度学习的背景下，负载不平衡是从始至终贯穿MoE应用的一个问题，该问题通常可以通过调整门函数、正则化、分配专家容量以及随机门函数，并配合额外的损失函数来解决。 MoE模型结构如图一所示。

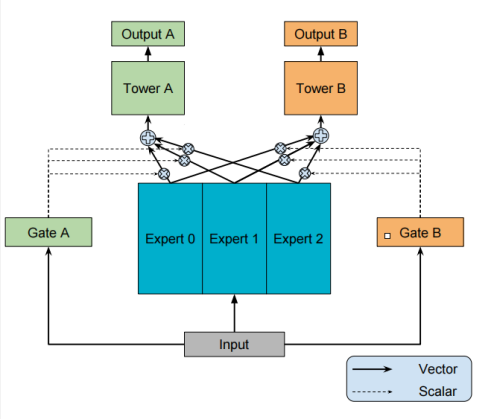


图一：MoE模型结构

多任务学习是基于共享表示，把多个相关的任务放在一起学习的一种机器学习方法。多任务学习涉及多个相关的任务同时并行学习，梯度同时反向传播，利用包含在相关任务训练信号中的特定领域的信息来改进泛化能力。多任务学习模型可以通过利用正则化和迁移学习来改善所有任务的模型预测。得到广泛应用的多任务模型是共享底层模型，该模型具有共享底层模型结构，底层隐藏层在任务之间共享。这种结构在很大程度上降低了过拟合的风险，但可能会遭受由任务差异引起的优化冲突，因为所有任务都需要在共享底层上使用相同的参数集。后来，研究员对该方法进行了改进，不是在任务之间共享隐藏层和相同的模型参数，而是在特定于任务的参数上添加不同类型的约束。与共享底层模型相比，这些方法具有更多特定于任务的参数，当任务差异导致共享参数更新冲突时，可以获得更好的性能。然而，大量的任务特定参数需要更多的训练数据来拟合，并且在大规模模型中可能不有效。多任务学习有很多的应用场景，可以做多语言机器翻译任务、推荐系统任务等。但是，多任务学习模型并不总是在所有任务上都优于相应的单任务模型，多任务学习模型对数据分布差异和任务之间的关系等因素很敏感。在相关性很高的多个任务中，单个模型表现的性能通常不是很好。在实践中，有一部分研究员为每个任务添加更多的模型参数，以适应任务差异，这样就造成模型参数的增长，对设备的要求变高。通常，相似的子任务也拥有比较接近的底层特征，那么在[多任务学习](https://so.csdn.net/so/search?q=%E5%A4%9A%E4%BB%BB%E5%8A%A1%E5%AD%A6%E4%B9%A0&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_41888257/article/details/_blank)中，他们就可以很好地进行底层特征共享；而对于不相似的子任务，他们的底层表示差异很大，在进行参数共享时很有可能会互相冲突或噪声太多，导致多任务学习的模型效果不佳。

基于物品的协同过滤是推荐系统中知名度最高的方法，但基于物品的协同过滤在面对数据稀疏的情况下效果急剧下降。同时基于物品的协同过滤倾向于推荐用户购买过商品的类似商品，往往会出现多样性不足、推荐惊喜度低的问题。基于用户的协同过滤方法在推荐结果的新颖性方面有一定的优势，但是推荐结果的相关性较弱，且容易受潮流影响而倾向于推荐出大众性物品。此外，隐语义与矩阵分解模型及其各种改进升级方法是推荐精度最好的单一模型方法，但当数据规模大时其运算性能会明显降低，同时方法依赖全局计算信息，因而很难作增量更新，导致实际工程中会遇到不少困难。基于内容的推荐算法是最直观的推荐算法，这种方法实现简单，应对的场景丰富。但推荐的效果不是很好。基于统计思想的一些方法，计算速度快，但是对用户个性化偏好的描述能力弱，实际应用时也存在各种各样的问题。以上是对推荐系统简单的技术介绍。

MMoE模型是学习从数据中建模任务关系的模型，显式地对任务关系建模，并学习特定于任务的功能以利用共享表示。允许自动分配参数以捕获共享任务信息或特定于任务的信息，从而避免了为每个任务添加许多新参数的需要。MMoE的主干是建立在最常用的共享底层多任务DNN结构之上的。该模型不是所有任务共享一个底层网络，而是有一组底层网络，每个网络被称为一个专家，每个专家都是一个前馈网络，且每个任务对应单独的一个门控网络。门控网络对于每个专家赋予不同的权重，使不同任务对专家的利用不同。然后，综合专家的结果，将其传递到特定任务的“tower”网络。这样，针对不同任务的门控网络可以学习不同的专家组合组合模式，从而获取任务关系。模型结构图如图二所示。



图二：MMoE模型结构图

该模型的输出计算公式为：



其中，



为输入特征，为可训练参数，为第i个专家的参数，为第k个任务的门控网络，为第k个任务的“tower”网络。

每个门控网络针对输入示例，可以学习选择一个专家子集使用。这对于多任务学习中灵活的参数共享是必要的。作为特殊情况，如果只选择一个门限得分最高的专家，每个门限网络实际上将输入空间线性分割为n个区域，每个区域对应一个专家。MMoE能够以一种复杂的方式对任务关系进行建模，方法是确定由不同门造成的分离如何相互重叠。如果任务不太相关，那么共享专家就会受到惩罚，这些任务的门控网络将学会利用不同的专家。MMoE只有几个额外的门控网络，门控网络中的模型参数数量可以忽略不计。因此，整个模型在多任务学习中仍然尽可能使用到知识迁移。

论文作者给出MMoE实验结果以及优点，MMoE具有可训练性、有效性，且计算效率高，可训练性即模型在超参数设置和模型初始化范围内的鲁棒性好。有效性指在解决由任务差异冲突引起的冲突时，MMoE模型更有效。

代码在TensorFlow Keras环境下运行，数据集为人口普查收入数据集，也就是论文中6.3小节所使用的数据集，数据集的列名有年龄、工作编号、时间、收入、婚姻状况等内容，共有40个特征。训练集为199523条数据，测试集和验证集共49881条数据，且比例为1：1。

使用代码表示公式2中的为：

expert\_outputs = tf.tensordot(a=inputs, b=self.expert\_kernels, axes=1)  
if self.use\_expert\_bias:  
 expert\_outputs = K.bias\_add(x=expert\_outputs, bias=self.expert\_bias)  
expert\_outputs = self.expert\_activation(expert\_outputs)

使用代码表示公式3为：

for index, gate\_kernel in enumerate(self.gate\_kernels):  
 gate\_output = K.dot(x=inputs, y=gate\_kernel)  
 if self.use\_gate\_bias:  
 gate\_output=K.bias\_add(x=gate\_output, bias=self.gate\_bias[index])  
 gate\_output = self.gate\_activation(gate\_output)  
 gate\_outputs.append(gate\_output)

使用代码表示公式2最终结果为：

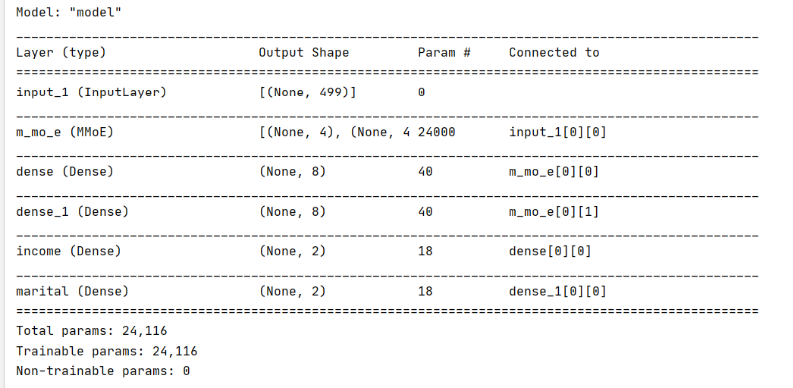
for gate\_output in gate\_outputs:  
 expanded\_gate\_output = K.expand\_dims(gate\_output, axis=1)  
 weighted\_expert\_output=expert\_outputs\* K.repeat\_elements(expanded\_gate\_output, self.units, axis=1)  
 final\_outputs.append(K.sum(weighted\_expert\_output, axis=2))

MMoE的“tower”网络实现由具有ReLU激活的相同多层感知器组成。门控网络只是输入的一个softmax层的线性变换，代码如下：

for index, task\_layer in enumerate(mmoe\_layers):  
 tower\_layer = Dense(  
 units=8,  
 activation='relu',  
 kernel\_initializer=VarianceScaling())(task\_layer)  
 output\_layer = Dense(  
 units=output\_info[index][0],  
 name=output\_info[index][1],  
 activation='softmax',  
 kernel\_initializer=VarianceScaling())(tower\_layer)  
 output\_layers.append(output\_layer)

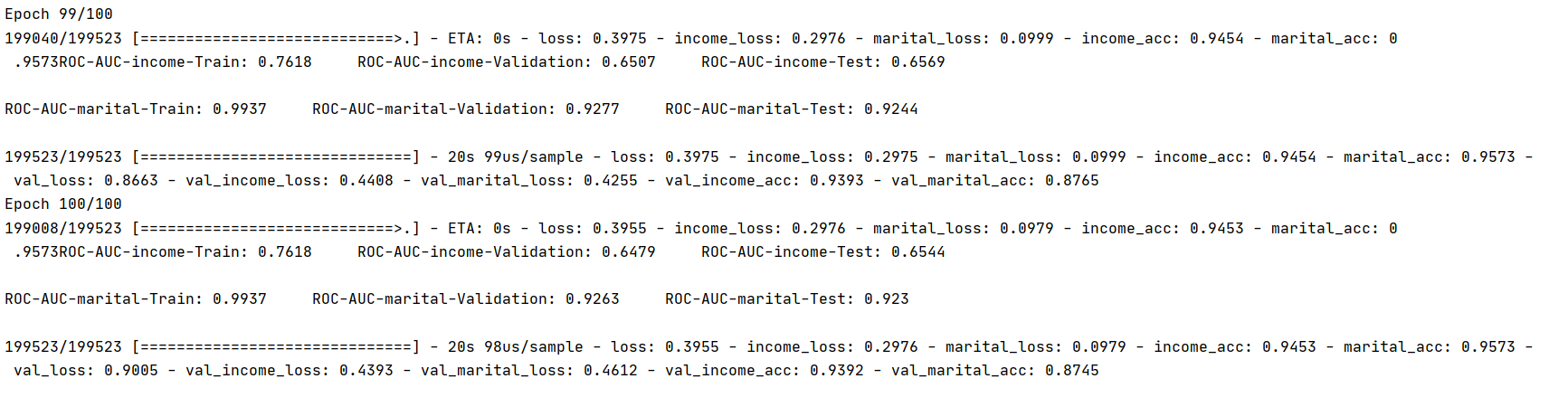
从该数据集中构建了多任务学习问题，任务1：预测收入是否超过$50K；任务2：预测这个人的婚姻状况是否从未结婚。由于该任务是二分类问题，所以使用AUC分数作为评估指标。对于超参数调优，使用验证集上收入任务的AUC作为目标。因为数据量不是很大，所以运行100个epoch就能取得很好的效果。

模型的结构及训练参数为：



图三：模型结构及参数

模型运行截图如图四所示。



图四：模型运行截图

项目文件中，Train-ROC为两个任务在训练集上的ROC值，Test-ROC为两个任务在测试集上的ROC值，Validation-ROC为两个任务在验证集上的ROC值，收入任务的训练集，ROC值经过100次训练后，从0.745变化到0.761，而婚姻状况任务的训练集，ROC值经过100次训练后，从0.987变化到0.993。

MMoE模型在很大程度上保留了计算优势，因为门控网络通常是轻量级的，专家网络在所有任务中共享。此外，该模型通过将门控网络作为稀疏结构来实现更好的计算效率。未来MoE模型将更多的应用在多任务学习上，或者是多语言任务、多领域任务上，用来提高计算效率和准确率。